KI891-0184 FG
B. H. Ratcliff

IBM® Technical Disclosure Bulletin Vol. 38 No. 09 September 1995

Software Counters for First-Failure Data Capture

Disclosed is a method for using counters imbedded inside of software programs to aid in debugging. By using software counters at various points in the normal software flow and in all error conditions, the past and present status of the system can be determined. The software counters are continuously updated as the system is running. When an error occurs or an abnormal path in the code is taken, the associated counter will be incremented. This keeps an eternal log of all system activity. This information is never overlaid or lost.

The software counters cause no performance degradation to the system and they are always functioning. When an error occurs for the first time, a substantial amount of data is available to determine the cause of the error. This greatly increases the odds of solving an error on its first occurrence. At worst, the system programmer will have very useful information to use in debugging the problem to determine the next step without having to run various traces for an undetermined amount of time and wait for the problem to recur.

The debug and first failure data capture methods for most products today involve the use of a trace facility. This trace facility requires a large memory area or file to store the trace data and also affects the performance of the system. On Local Area Network (LAN) adapters, this becomes a problem because of the lack of storage and the performance impact of running the trace.

A major problem on many systems has been the gathering of useful information on the first occurrence of a failure. This is known as first failure data capture. Most trace facilities are not run during "normal" production periods because of the performance impact. Therefore, on the first occurrence of an error, no information is available.

A second problem is the wrapping of the trace table after a failure occurs. Many times problems are intermittent and the initial failure is not easily detected at the system level. By the time a user realizes the problem has occurred and tries to save the trace table, the trace table has wrapped and all relevant information about the error is lost. This is especially true on small systems like LAN servers and adapter microcode.

Software counter example - In the 3172, software counters were used for debugging when the performance of trace was too degrading and masked the error from occurring. The specific function in the 3172 where these software counters were used was in the transmission and reception of frames of the Fiber Distributed Data Interface (FDDI) media.

In the transmit flow, four steps were involved. The first step was the notification by the HOST operating system a frame was ready to be sent on the fiber. The second step was th moving of data from HOST memory to the FDDI shared memory. The third step was the moving of the data from the FDDI shared memory to the FDDI chipset memory. The fourth and final step was to notify the FDDI chipset a frame was ready to be transmitted. At each of these steps it was possible for the HOST frame to be lost because of various reasons ranging from lack of storage to FDDI chipset errors. A software counter was used at each step to track the progress of the frame. For all possible error conditions in the transmit flow, a software counter was also used. These software counters gave the ability to continuously monitor the system activity. At any one point in time, it was easy to determine if an abnormal number of frames were being lost, data errors were occurring or the system was performing normally.

The receive flow was also performed in four basic steps. As with the transmit flow, the received frame could be discarded or lost at any one of the steps because of an abnormal error condition. Software counters were placed at each stage of the received process and at all abnormal paths.

After fully implementing the use of software counters throughout FDDI microcode, they were used for debugging any problems which were masked by the trace utility. The use of system trace was no longer necessary. The software counter immediately identified the code path which was in error.

The advantages of software counters follow:

Continuously Running: The software counters are compiled in line and are incremented each time the code section is executed. Periodic "snap shots" of the system can be taken to monitor the overall system activity and performance. There is no user interaction which can result in the wrong data being captured.

No performance Degradation: Unlike system traces, the software counters induce no degradation to the system performance. No file writes or text conversion is necessary. A simple increment of a memory variable is all the occurs.

First Failure Data Capture: Whenever an error occurs, the software counter information is present and can be extracted to check for any abnormal counter increments. Also, all events are being tracked and many errors occur when abnormal code paths are executed. By having the software counters, it is always possible to determine which abnormal code path was executed. With system traces, it is far too expensive to run a trace to cover all possible paths. For many errors, the system programmer never has any idea what the root cause of an error was. Sometimes special traces and traps must be written if system traces fail to capture the cause of the error. Software counters eliminate the system programmer time spent determining the root cause of an error. This allows him to spend more useful time with the customer.

System and Unit test verification: By using software counters, various test cases can be verified to execute specific code paths. The problem with many test cases is not knowing if all various code paths are tested. With software counters present, it is obvious if a code path was executed or not.

Small Area of memory required: Each software counter is a 32 bit counter which requires 4 bytes of memory. To implement 256 specific counters only requires a 1K block of memory. Most useful trace tables are over 32K in size. Many other require a large file to be allocated on a disk which adds greatly to the performance impact of the trace.

Below are disadvantages of the software counters:

To fully utilize the information made available by the software counters, a programmer must have a good understanding of the codes function and flow. Many counters can be used to track the flow of data through a task. If the programmer is not familiar with the specific task, the software counters will not yield any useful information.

When new counters need to be added, the code must be recompiled and the structure variable which contains the counters must be changed. This requires a new level of code to be sent when a new counter is added. However, the same basic problem exists with system traces today. When a new trace event needs to be added, the code must be recompiled.